Support Infrastructure for Application Layer Protocols

Roch Glitho, PhD

Full Professor

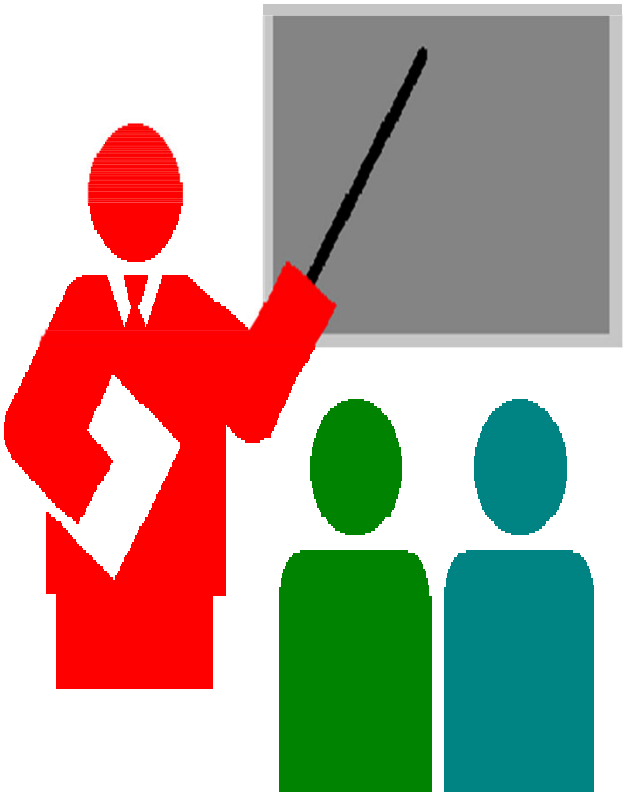
Ericsson / ENCQOR-5G Senior Industrial Research Chair

Cloud and Edge Computing for 5G and Beyond

My URL - http://users.encs.concordia.ca/~glitho/

Concordia University
Engineering and Computer Science
Concordia Institute for Information Systems Engineering

Support Infrastructure



- Distributed Name Service (DNS)
- Peer to Peer

Chapter VII
Support Infrastructure for Application Layer: Distributed Name System DNS)

Support Infrastructure

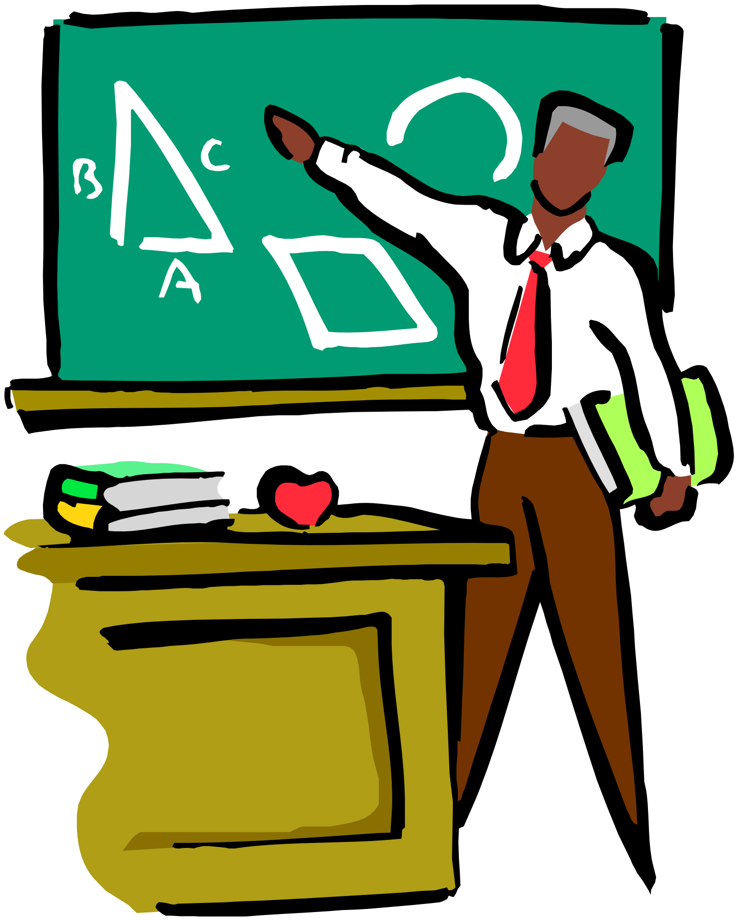
Support infrastructure for application layer

- Why?
  - Re-usability across application layer protocols
  - Modularity (i.e. separation between application layer protocol specification / design and infrastructure specification / design)

Support Infrastructure

- Support infrastructure for application layer
  - Examples discussed in this course
    - Distributed Name System (DNS)
      - Mapping between application layer symbolic addresses and IP addresses
    - Peer to peer overlays
      - Connectivity, routing and messaging between peers for applications such as file sharing, IP telephony

Domain Name System



- 1 – Conceptual Framework
- 2 - Implementation architecture

Conceptual Framework

Genesis

- Early 80s
- Replacement of the HOSTS.TXT file used in the early days of the Internet
  - HOSTS.TXT file
    - Contains list of all Internet host and mapping between symbolic names and IP addresses
    - Centrally managed and periodically distributed to all Internet hosts via file transfer

Conceptual Framework

Genesis

- Not sustainable with Internet growth
  - Scalability issues
  - Administrative issues (e.g. who will store and maintain this critical file?)

Conceptual Framework

Design goals / requirements

- Provide as a minimum all information provided by HOSTS.TXT
- Distributed storage and maintenance
- Flexible syntax for names and sizes of data associated with names
- Inter-operability across Internet
- Tolerable performance
- Extensibility
- Independence of network topology and operating systems

Conceptual Framework

Design goals / requirements

- Examples of implied design decisions
  - **Hierarchical name space**
    - Distributed storage and maintenance
    - Flexible syntax for names and sizes of data associated with names
  - **Data caching whenever possible**
    - Tolerable performance
  - **Lean solution vs. comprehensive solution**
    - Extensibility

Conceptual Framework

Key concepts

- Domain name space
  - Variable depth tree with labelled nodes
    - Labels
      - » Variable length strings of octets
      - » Case insensitive

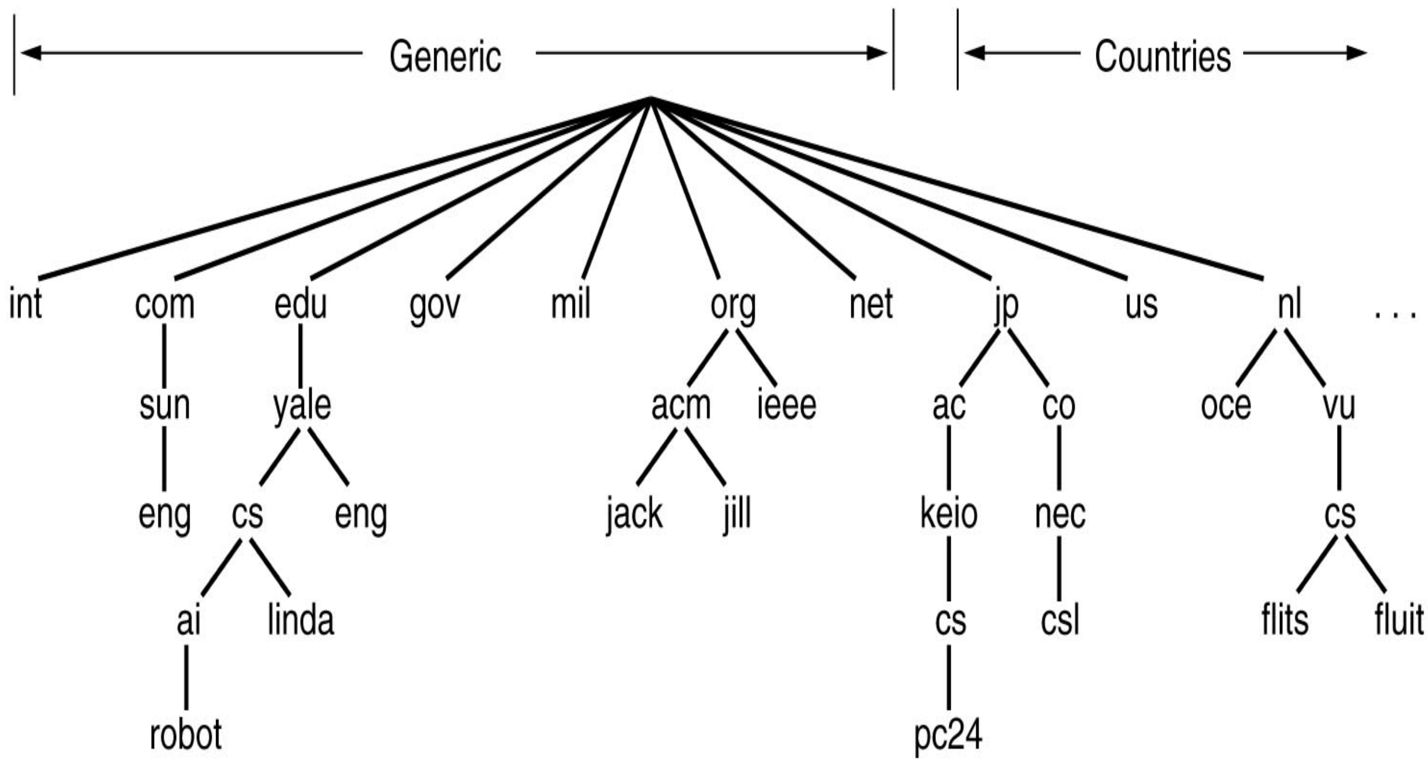
Conceptual Framework

Key concepts

- Domain name of a node
  - Concatenation of all labels from the node to the root
- Administrative decision
  - Top levels correspond to:
    - Country codes
    - Broad organization types

Conceptual Framework

Domain name space



Conceptual Framework

Key concepts

- Resource records (RRs)
  - Data attached to each name
    - Type
      - » Abstract resource (e.g. host addresses)
    - Class field
      - » Protocol family (e.g. DARPA Internet)
    - Application data
  - Authoritative record vs. cached record
    - Cached record may be out of date

Conceptual Framework

Key concepts

– Resource records (RRs)

Type	Meaning	Value
SOA	Start of Authority	Parameters for this zone
A	IP address of a host	32-Bit integer
MX	Mail exchange	Priority, domain willing to accept e-mail
NS	Name Server	Name of a server for this domain
CNAME	Canonical name	Domain name
PTR	Pointer	Alias for an IP address
HINFO	Host description	CPU and OS in ASCII
TXT	Text	Uninterpreted ASCII text

Conceptual Framework

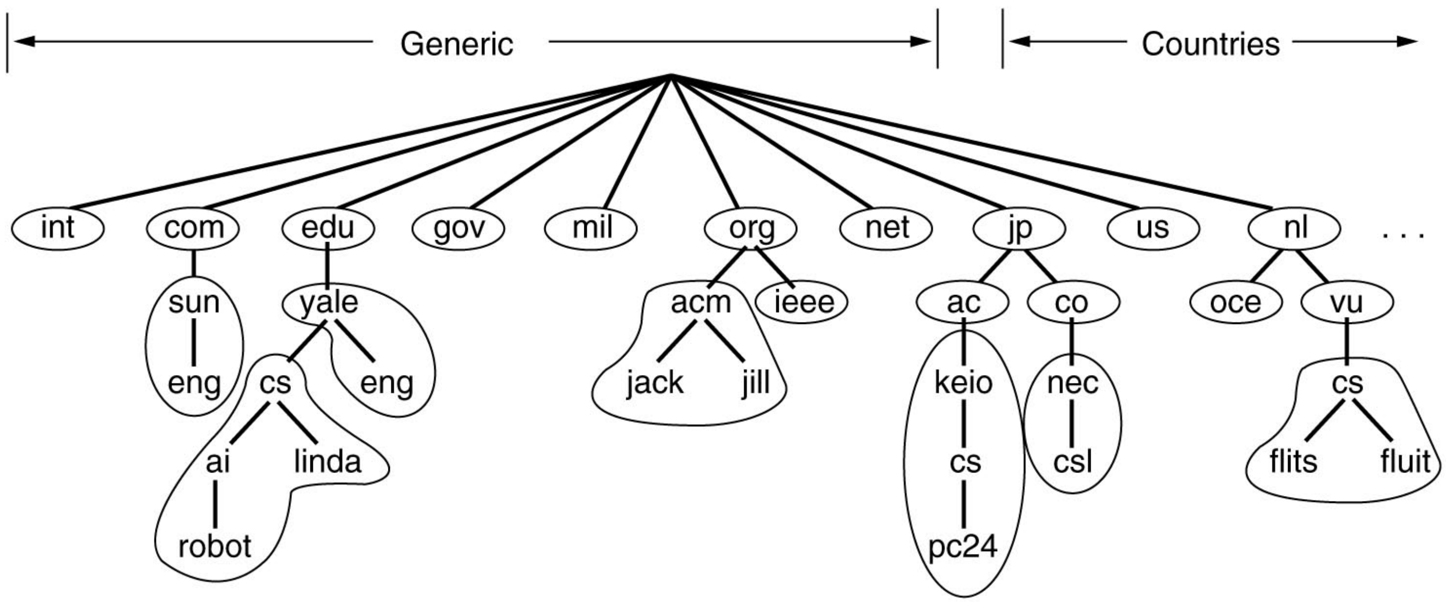
Key concepts

- Zones
  - Non overlapping sub-trees under the control of a single organization
    - Single nodes and whole tree are excluded
  - Procedure
    - Initial authorisation obtained from a parent organization for a single node
    - Growth to an arbitrary size without involving the parent organisation
  - Organization responsibilities
    - Maintenance of zone data
      - » Ensure reliability through redundancy

Conceptual Framework

Key concepts

– Zones



Conceptual Framework

Key operations

- Zone transfers
  - Organizations make their zones available throughout the Internet
- Caching
  - Organizations cache whenever possible the zones they received from other organizations

Implementation Architecture

- Client – server implementation
- Functional entities
  - Name server
    - Information repository
    - Does the actual mapping (i.e name resolution)
    - Can support any number of zones
      - Flexibility
        - » A name server for a given zone does not need to be in the zone
        - » Optimal distribution when distribution follows name space hierarchy

Implementation Architecture

Functional entities

- Resolver
  - Interface to client programs
    - Implement algorithms for finding the name server that has the required mapping information

Implementation Architecture

Implementation considerations

- Name server and resolver may be in separate boxes (nodes) or in a same node
  - Resolver is usually centralized in dedicated servers at organization levels
    - Re-use of cached information
    - No need for less powerful machines to implement their own resolving function

Implementation Architecture

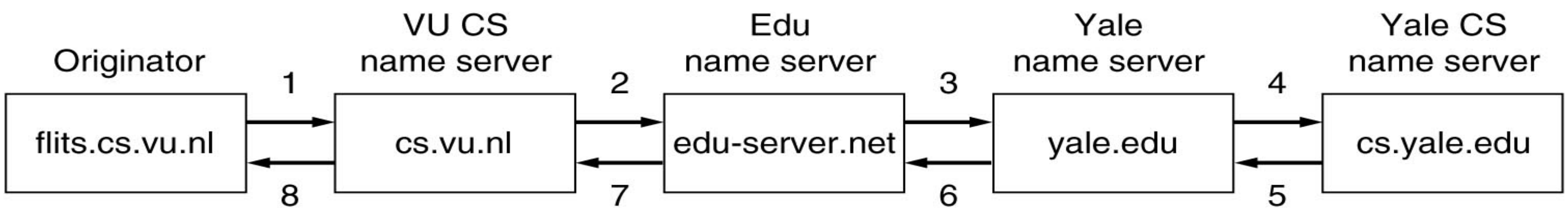
Implementation considerations

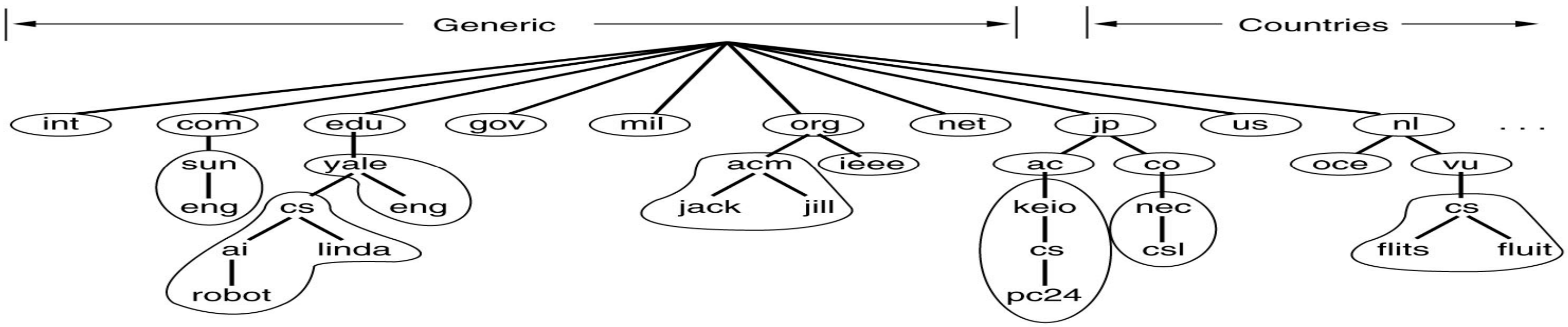
- Queries
  - Two approaches
    - Recursive (optional)
      - » When a name server does not have the requested information, it tries directly a name server that may have it
      - » Process is iterated till information found
    - Iterative
      - » When a name server does not have information, it returns to the resolver the name of the next name server on the path

Implementation Architecture

Implementation considerations

– Example of recursive look up (linda@cs.yale.edu)





Implementation Architecture

Implementation considerations

- Messages
  - Single format (Valid for both request and reply)

```
+--------------------+
|      Header        |
+--------------------+
|     Question       | the question for the name server
+--------------------+
|      Answer        | RRs answering the question
+--------------------+
|     Authority      | RRs pointing toward an authority
+--------------------+
|    Additional      | RRs holding additional information
```

Implementation Architecture

Implementation considerations

- Transport
  - UDP
    - Queries
      - » Eventual retransmissions left up to applications (i.e. resolver and name server)
        - » Guidelines provided
  - TCP
    - Zone related information (i.e. refresh)

Implementation Architecture

Problems with current implementation

- Backed by experience and experimentation
  - Vulnerability to network failures and Dos
    - » Key reasons:
      - » small number of name servers and limited redundancy
      - » Known vulnerabilities in commonly deployed name servers
  - Performance issues
    - » Name resolution latency
      - » Reasons:
        - » Low cache hit
        - » Human errors (i.e. misconfigurations)

Implementation Architecture

Research on alternatives

- Rely mostly on P2P design instead of client / server design
  - Did not really fly

References

1, P.V. Mockapetris and K. Dunlap, Development of the Domain Name System, ACM SIGCOMM Computer Review, 1995

2. P.V. Mockapetris, RFC 1034 and RFC 1035, November 1987
3. A. Tanembaum, Computer Networks, Chapter 7
4. V. Ramasubramanian and E. Sirer, The Design and Implementation of a Next Generation Name Service for the Internet, SIGCOMM'04, August 30 – Sept. 3,2004

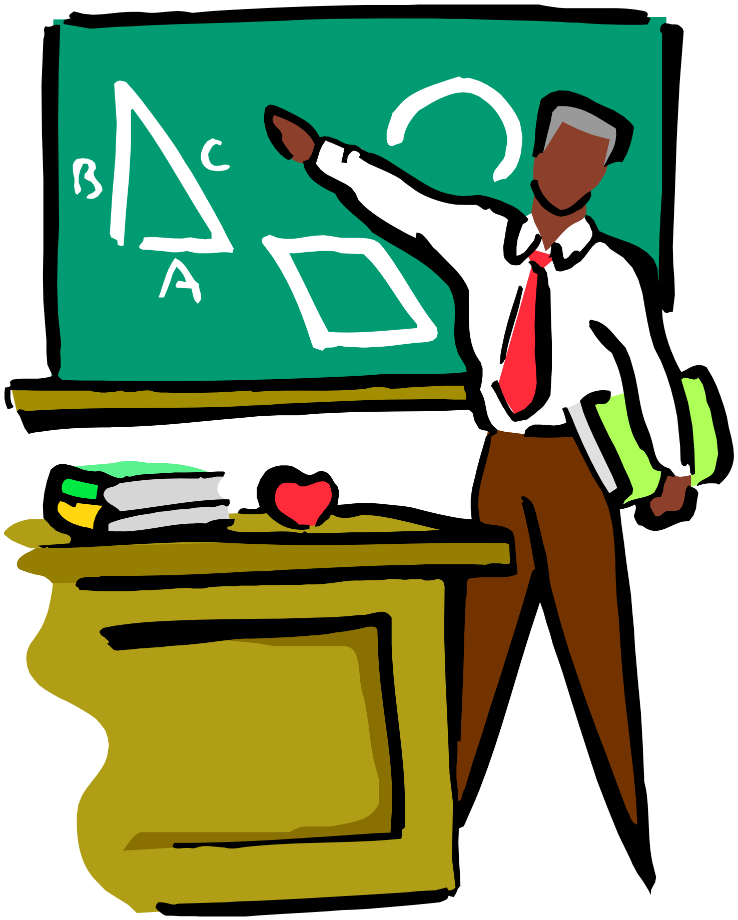
Chapter VIII
Support Infrastructure for Application Layer: Peer-to-Peer (P2P) Overlays

Support Infrastructure

Support infrastructure for application layer

- Why?
  - Re-usability across application layer protocols
  - Modularity (i.e. separation between application layer protocol specification / design and infrastructure specification / design)
- Examples discussed in this course
  - Distributed Name System (DNS)
    - Mapping between application layer symbolic addresses and IP addresses
  - Peer to peer overlays
    - Connectivity, routing and messaging between peers for applications such as file sharing, IP telephony

P2P Overlays



- 1 – Client/server vs. P2P computing
- 2 - Structured P2P Overlays vs. Unstructured P2P Overlays
- 3. Chord, Freenet and Skype
- 4. JXTA: A middleware for P2P applications development

Client / server vs. P2P computing

Client / server

- Essence
  - Single server offering storage and computation
    - Static
    - Updates done solely by server provider
  - Clients access server
    - Passive role
    - No (or little) contribution to storage and computation
- Underlying assumption
  - Clients have no (or little) storage and computation capabilities

Client / server vs. P2P computing

Client / server

- Inherent issues:
  - Root:
    - server is a computational and network bottleneck
      - » Examples of issues
        - » Scalability
        - » Availability
        - » Efficiency

Client / server vs. P2P computing

P2P computing

- Several possible definitions
  - In this course:
    - Computing paradigm that relies on a network of peers (instead of a server) to solve the issues inherent to client / server paradigm, such as:
      - » Scalability
      - » Availability
      - » Efficiency
- Underlying assumption
  - Clients now have more and more storage and processing power that should be used

Client / server vs. P2P computing

P2P computing

- Clients federate via a P2P network to offer storage and computational capabilities required by applications
  - Clients are called peers
    - Each peer may contribute according to its capabilities
    - More powerful peers sometimes called super – peers may contribute more

Client / server vs. P2P computing

P2P computing

- Pure P2P vs. hybrid P2P
  - Pure P2P:
    - » Fully decentralized architecture
      - » Not that common (e.g. Freenet)
  - Hybrid P2P
      - » Some level of centralization
        - » More common (e.g. Skype)

Client / server vs. P2P computing

P2P computing

- Some examples of technical challenges
  - Self organization
    - Peers may join or leave the network anytime
  - Storage and look up
    - Where to store?
      - » Items may be stored on any set of peers
    - Efficiency of lookups (guarantee vs. performance)
  - Fault tolerance
    - Voluntary departures vs. un-voluntary departures
      - » What to do if a peer leaves?
      - » What to do if a peer goes down?

Structured P2P overlays vs. unstructured P2P overlays

P2P overlay

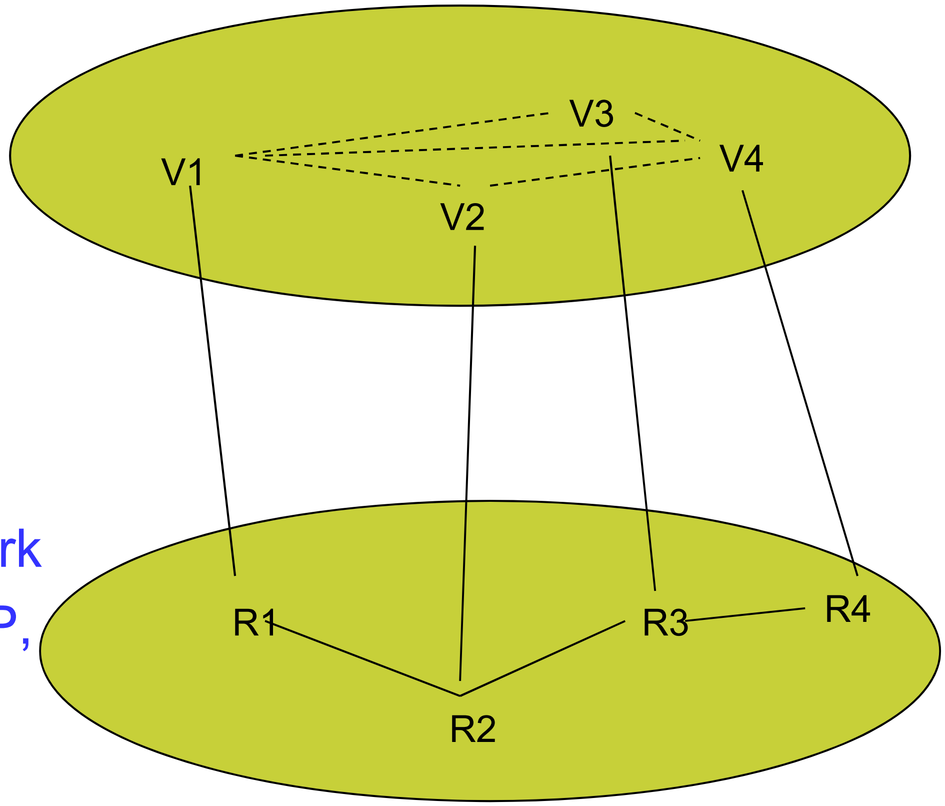
- Current way of implementing P2P computing
  - Application layer virtual networks that provide storage, processing, connectivity and routing
    - Network built by peers that federate to offer storage and processing capabilities to applications
      - » Built on top of existing networks, thus the name of overlay
        - » Applications running on top of transport protocols of real network
        - » Real network nodes become virtual nodes in the overlay

Structured P2P overlays vs. unstructured P2P overlays

P2P overlay

Overlay (Above Transport)

Real network (PHY,link,IP, Transport)



Structured P2P overlays vs. unstructured P2P overlays

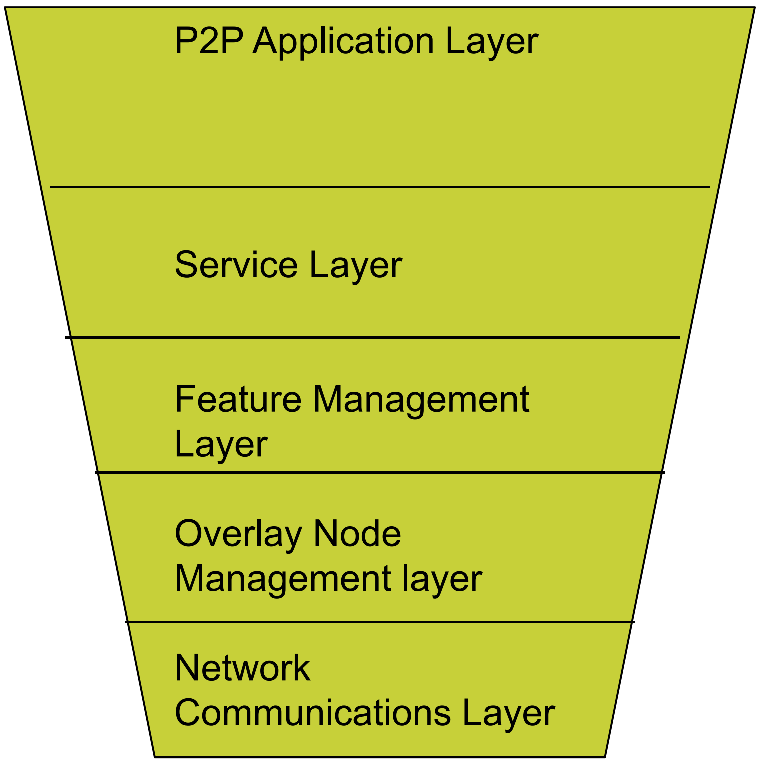
P2P overlay

- Characteristics
  - own topology that may be different from the topology of the real network
  - Own protocols that may be different from the protocols used in the real network
  - May come with an application embedded in it (e.g. Skype) or as an infrastructure that can be used by other applications (e.g. CHORD)
  - APIs, toolkits are provided when the application is not embedded in the overlay

Structured P2P overlays vs. unstructured P2P overlays

P2P overlay

Simplified abstract view (All this is above transport)



Structured P2P overlays vs. unstructured P2P overlays

Simplified abstract view

- Application layer
  - Actual P2P applications (e.g. file sharing, IP telephony)
    - » Maybe either embedded in the P2P infrastructure or built by developers using APIs depending on the P2P overlay
- Service layer
  - Services or building blocks used by developers to build applications
    - » Maybe or may not be visible to third party developers
- Feature management
  - • Features used by all applications (e.g. security, fault resilience)

Structured P2P overlays vs. unstructured P2P overlays

- Overlay nodes management
  - Routing, resources discovery, location look up
- Service layer
  - Services or building blocks used by developers to build applications
    - » Maybe or may not be visible to third party developers
- Network communication layer
  - • Interface to the real network
    - On top of a real transport network

Structured P2P overlays vs. unstructured P2P overlays

Structured P2P overlays

- Tightly controlled topology
  - Content placed at very specific locations
    - Efficient subsequent queries
    - Technique used: Distributed Hash Table (DHT)
      - Generation of a key
        - » Put (Key, value)
        - » Value = Get (key)
      - Each peer has a small routing table of neighbouring peers
        - » Node ID
        - » IP address
      - Messages routed progressively using Node ID that are closer to the key

Structured P2P overlays vs. unstructured P2P overlays

Structured P2P overlays

- Some examples
  - Content Addressable Network (CAN)
  - Chord
  - Tapestry
  - Pastry
  - Kademlia
  - Viceroy

Structured P2P overlays vs. unstructured P2P overlays

Un-structured P2P overlays

- Loosely controlled topology
  - Content placed at random locations
    - Flooding techniques
      - » Efficient for highly replicated content
      - » Inefficient for rare content

Structured P2P overlays vs. unstructured P2P overlays

Un-structured P2P overlays

- Some examples
  - Napster
  - Freenet
  - Gnutella
  - KazaA
  - BitTorrent

Structured P2P overlays vs. unstructured P2P overlays

Chord, Freenet and Skype

- Chord
  - Structured P2P overlay that can be used to build applications
  - An example of possible applications
    - Time shared storage for nodes with intermittent connection
      - Goal:
        - » Have one's data always available (even when disconnected)
      - Solution
        - » Store the data of other peers when connected and get ones data stored by other peers when disconnected

Structured P2P overlays vs. unstructured P2P overlays

Chord, Freenet and Skype

- Chord
  - Key features
    - Load balancing
    - Full decentralization
    - Scalability
    - Availability
    - Flexible naming

Structured P2P overlays vs. unstructured P2P overlays

Chord, Freenet and Skype

- Freenet
  - Goal
    - Create an un-censorable and secure global information storage system
      - Unstructured P2P
      - Application embedded in the P2P overlay
        - » Efforts to decouple the application from the P2P overlay were not successful

Structured P2P overlays vs. unstructured P2P overlays

Chord, Freenet and Skype

- Freenet
  - Requirements
    - Privacy for information producers, consumers and holders
    - Resistance to information censorship
    - High availability and reliability
    - Efficient, scalable and adaptive storage and routing

Structured P2P overlays vs. unstructured P2P overlays

Chord, Freenet and Skype

- Freenet
  - Architectural principles
    - Users use globally unique identifier (GUID) to insert and retrieve files
      - GUIDs are assigned by the system
      - Data may be encrypted before insertion in the network
      - Files are stored on some set of nodes (may migrate or be replicated)
    - Messages travel through node to node chains and each link is individually encrypted
    - Controlled fllooding

Structured P2P overlays vs. unstructured P2P overlays

Chord, Freenet and Skype

- Skype
  - Application embedded in the P2P overlay
    - No design information available in the public domain
      - The little that is known is by reverse engineering

Structured P2P overlays vs. unstructured P2P overlays

Chord, Freenet and Skype

- Skype
  - Several servers
    - Login server
    - Skype-out server (PC to Public Switched Telephony Network (PSTN) calls)
    - Skype 0 in server (PSTN calls to PC)

Structured P2P overlays vs. unstructured P2P overlays

Chord, Freenet and Skype

- Skype
  - Several servers
    - Login server
    - Skype-out server (PC to Public Switched Telephony Network (PSTN) calls)
    - Skype - in server (PSTN calls to PC)

Structured P2P overlays vs. unstructured P2P overlays

Chord, Freenet and Skype

- Skype
  - Overlay architecture
    - Hierarchical
      - Ordinary host
      - Super nodes
        - » Every ordinary host is connected to a super node
          - » Routing table contains list of reachable super nodes
        - » Super nodes are interconnected

Structured P2P overlays vs. unstructured P2P overlays

Chord, Freenet and Skype

- Skype
  - Signaling
    - Always over TCP
      - May go directly from caller to callee (if callee is in caller busy and both are with public IP)
      - May go via a super node (When for instance callee is behind a NAT)

The End